# JAVA PROJECT - QUIZ MANAGER

EPITA

ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

Submitted By :-

Nitesh Kumar Jha

Kannan Veerabhagu Sathanam

## Acknowledgement

The success and outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We respect and thank Quentin Decayeux and Thomas Broussard for providing us an opportunity to do the project work and giving us all support and guidance, which made us complete the project duly. We are extremely thankful to them for providing such a nice support and guidance.

## Table of Contents:

## Introduction

This project is made to let the students take the quiz and it would then evaluate the result of the quiz for each test using the unique login name or login ID of the student.

This quiz would be stored/uploaded by the admin using his credentials and then this quiz would be available to the students.

Students can register before taking the test or login using their login name and password.

Each user would have the unique ID by which their result would be evaluated

We have used H2 database to store and interact with the application to store, display questions based on the difficulties and topics and the choices to the students and storing the result of the quiz.

Questions would be displayed one by one to the student and student need to answer the questions to go to the next questions and it would be stored in the database simultaneously after each question being answered

We have used CRUD operations for MCQ and Open questions which would store the questions and valid answers would be store in database.

These questions can be searched using the topics and difficulties and result for the quiz would be shown at the end of the quiz.

We would use GUI for the user to take the quiz where user would get option to login and then using the GUI user would be able to go to different/next questions and answer based on the questions like if it's an open question or Multiple-Choice Question.

The user guide would be made available with the steps and screenshot to help the admin as well as student to update and take the test.

Graphical Interfaces has been created for the students and admin to login to their account where admin can setup the quiz and student can login or sign up and take the quiz.

When student or admin would open the application, it would show the Home Page where user would get the option to select "Student" or "Admin"

Selecting students would give students two option, Signup or Login. Student would be able to take the quiz after successfully login using it's credential.,.

Similarly, when admin clicks on the "Admin", it would take it to the admin page and then the admin would be able to login and set the quiz.

## Different Packages and It’s corresponding Classes:

***Demo Package:-***

**Admin_in:**

This is to set the interface once the user is logged in and to provide the options with Create, Update, Delete.

**Admin:**

This is to present the interface to the admin to login to the application. This would validate the user's account and let the user access if the username and password matches. It would also be able to handle the exceptions

**Create:**

When user would Select Create option from the admin panel, user would get interface to create a question, set ID, set difficulty, topic etc

**Delete :**

This is to delete the questions from the list of the questions by the admin using the interface. Once deleted this is setup to give the pop of message stating ‘The Question is Deleted Successfully’

**Home :**

This is the Home page.

It has the interface to Login for Admin and Student. It would show two option to click on (Student and Admin)

**Student_In:**

Once logged in, it would give user the user interface to select the topic, difficulty to choose from and proceed with the text.

**Student:**

This would provide used with the interface to Sign up or login.

This would also validate student's username and password and if it's wrong, give the error message.

Or If the student chooses to create a new account then once filled the information, it would give a pop up saying “Details Added”

This would also handle any expected exceptions that might occur

**Student_Question:**

It is to show the user with the questions with the options to choose from and then hit submit to submit the answer and see next questions.

**Update :**

This interface would give admin user to update existing questions. User can update ID, Question, Difficulty and Topics etc using this interface.

## Datamodel Package:

**Answer.Java :**

It contains all the answer associated with each question corresponding to the Questions' Id

**Exam.class :**

We have created this class to get and set the result for the student by evaluating the number of correct answers student has given

**MCQAnswer :**

This class would show the list of choices for the multiple-choice questions.

**MCQChoice:**

Using getter and setter method to set the label, assign the IDs to the questions and set the MCQ question.

**MCQQuestion** :

This class is created to inherit all the method of question class

**Question:** This class has again using getter and setter method to set the difficulty level of the questions and set different id's so that it can be used to evaluate the grading.

**Quiz:**

This would set and display the list of MCQ questions

**DAO:**

Choice File DAO: This would assign the 4 choices of the question asked according to their respective ids

It would also show the exceptions if there would be any while processing or assigning

**ChoiceFileDAO**

This would assign the 4 choices of the question asked according to their respective ids

It would also show the exceptions if there would be any while processing or assigning

**QuestionFileDAO**

This would store all the question stored in the file and loop over the lines

It would look for the question ID and label

Add the question to the final list.

**QuestionJDBCDAO**

This class would insert, Update, Delete and Search the questions.

It would also handle the exceptions while creating these operations with the database

**QuestionsXMLDAO:**

This would convert the questions from the XML file and set in based on id difficulties and topics

It would also let the admin update the questions and show the updated questions to the students

**Configuration:**

This would handle IO Exceptions

**Fr.epita.logger Package**

**Logger .java**

This is to connect the application with the H2 database using the below details

jdbc.url=jdbc:h2:tcp://localhost//C:/db/h2DS;create=true

jdbc.username=sa

jdbc.password=

**Fr.Epita.Services Package**

**Configuration :**

This is the configuration singleton class to initialise it for the first time calling getInstance

**Fr.Epita.Services.dao Package**

**ChoiceFileDAO**

It is to store the value of the choice made by the user in the database to evaluate the result

It is also setup to handle the expected exceptions that might occur

**QuestionFileDAO**

It is to get all questions stored into the file and add the question to the final list before creating another.

**QuestionJDBCDAO**

It is to interact with the database to CRUD (Create, Read, Update and Delete) operation for the questions and updating it's corresponding data like difficulty, topics, ID